Integrating Cactus Simulations within Triana Workflows

Tom Goodale
Center For Computation and Technology,
Louisiana State University

Ian J. Taylor and Ian Wang
School of Computer Science,
Cardiff University

Abstract

In this paper, we give a status report for an architecture that can be used to view or analyse the output of scientific applications within secure distributed environments. The model satisfies essential criteria for such an environment, such as traversing firewall issues, by constructing a one-way protocol for communication to clients wishing to view the progress of such an application. Our prototype uses the Cactus thorn architecture to implement the protocol, but we have already developed a stand alone version for use with legacy applications. The protocol uses Web service standards for implementation and deploys these within a dynamic environment enabling spontaneous run-time connectivity with the legacy codes running on the distributed resource. We demonstrate this capability in this paper by showing how Triana can be used to view the output of a Cactus simulation of a 3D scalar field produced by two orbiting sources.

1 Introduction

Simulating physical phenomena on computers involves increasingly complex workflows, involving a plethora of tools; most scientists and engineers perform each step of these workflows manually, involving ad-hoc procedures and much labour. In recent years there has been a move to automate this with environments in which the person running the simulation can connect the tools used for stages of the workflow together, these can be graphical in nature, such as those based upon the Common Component Architecture [1], e.g. SCIRun [2], or lower level, such as Condor [3]; a brief survey of some such tools appears on the Scientific Workflows Survey website [4]. The growth of the Grid has brought new challenges and new opportunities for the development of such tools.

In parallel, much of the underlying Grid middleware has advanced from an experimental or prototype state to become mature, stable and widely deployed e.g. [5, 6, 7]. These new platforms create a homogeneous working environment across a vast range of heterogeneous resources and are based on open middleware projects, such as Globus [8] and the VDT [9]. These evolving environments enable a vast distributed resource that can be utilised by a wide range of scientific applications including high-throughput cases that can be executed repeatedly across the resources by exploring different starting conditions or parameter sweeps. One example of such an application is Cactus [10], which has a generic architecture that can be used to develop a wide range of scientific applications.

In this paper, we present a protocol which we have developed which enables the distributed notification and uploading of files created by an application during the iterative time steps of a simulation; this allows existing simulation codes to be easily integrated into workflow tools with little or no modification. A prototype of this protocol has been demonstrated recently in SC2004, where we showed the visualisation of a 3D scalar field produced by two orbiting sources. This was accomplished by using this protocol to connect two independently developed frameworks - Cactus, running on an HPC resource, and Triana [11] running on a users workstation. Triana received notifications of the files created by Cactus and then selected the ones it wished to visualise. The result was that the user could see real-time JPEG images from the remote application, representing the 3 dimensions of the scalar field, as the simulation progressed. The specific scenario we aimed to address is illustrated in Figure 1. The steps are as follows:

Startup: the Cactus application is launched on the distributed resource. This could be accomplished in a number of ways e.g. using GRAM [12], condor [3], more rudimentary tools, such as SSH, or higher level tools which select from the available options, such as the GAT [13].

Web Service Initialisation: Triana, wishing to receive files from Cactus initialises and dynamically launches a Web service, using WSPeer[14], to receive file notification and the incoming data from the application.

Discovery: Cactus discovers the address (endpoint) of the web service which wishes to visualise or analyse its out-

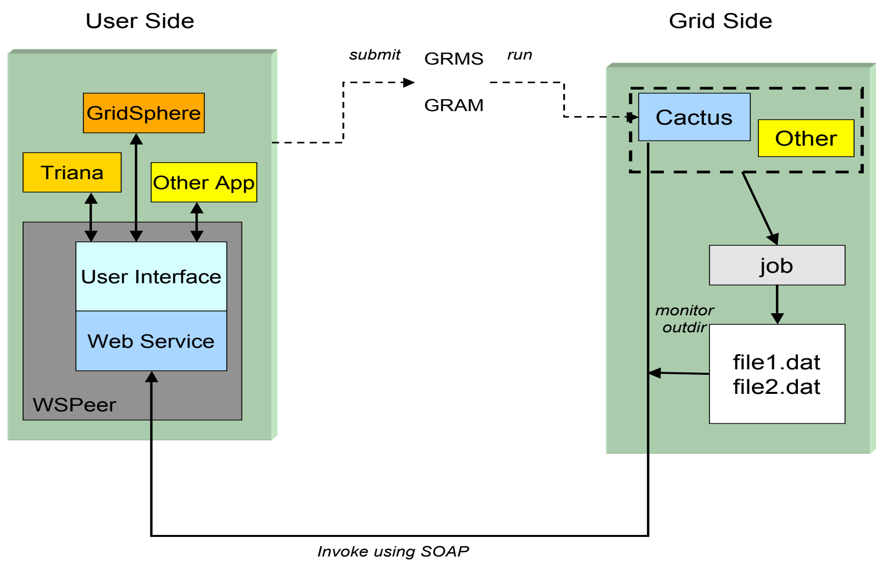

Figure 1: The architecture of the application monitoring subsystem employed within our workflow.

put and tries to connect. This is achieved through specified Unicast addresses at present but could be extended to utilise some kind of caching e.g. UDDI, or similar.

Notification: When Cactus outputs files it uses the protocol to notify the web service of the details, e.g. names and MIME types, of the files that are being created.

Selection: Triana decides which files it is interested in. Meanwhile, the Cactus application invokes the notification method on the web service. The web service returns the list of files it wishes to receive.

File Communication: Cactus then uploads the selected files and continues with the next time step of the simulation

Iterate: Cactus returns to the *selection* step upon completion of the next time step.

Figure 1 shows this scenario in a more generic sense. Here, we show how Cactus and Triana use of this protocol. Other client examples could include portal infrastructures or other visualisation applications and other Grid side application could include any legacy application that can run on a distributed resource. Since applications communicate through standard ASCII/binary files, there are no application requirements, other than that they generate some kind of files. Our prototype however, implements this framework within the Cactus application directly, although we have developed an independent monitoring tool, which exhibits the same characteristics.

The rest of the paper is organised as follows. First, the endpoint applications are briefly described; that is, Triana and Cactus. These technologies are fed into our scenario in section 4 by outlining the user perspective. We then describe our operating environment and address the indigenous firewall and NAT issues. We finally give a status report and outline future work on this project. We do not address the full life cycle of the workflow, such as how the Triana network and the Cactus simulation are instantiated, although such deployment issues are a crucial part of any grid infrastructure, they do not have a direct bearing on the protocol we discuss; one of the strengths of this protocol is that it allows two independently deployed instances of these frameworks to be used collaboratively, requiring no inter-dependence.

2 Triana

Triana is a graphical Problem Solving Environment (PSE) for composing scientific applications. Applications within Triana are created by dragging programming components, called tools, from the toolbox onto a workspace, and then drawing cables between these components to create a workflow. The original tools used within Triana were Java components run on the local machine, and a large suite of Java tools exist in a range of domains including signal, image and text processing. The signal processing tools are the most advanced as Triana was initially developed for data analysis within the GEO600 project [15].

Triana has been extended for use as a Grid Computing Environment (GCE), using the dynamic discovery and choreography remote services, such as Web Services, to extend its range of functionality. A service discovery interface allows remote service instances to be discovered through querying a directory service, such as UDDI, and the imported into the Triana toolbox. These remote components can be included into workflow alongside Java components, with the Java components generally being used to provide the input to and visualize the output from the remote services. A pluggable architecture allows additional component types and discovery mechanisms to be added easily.

Triana is also a test application for the GridLab project[16] and supports Grid job submission of legacy codes using the GAT [13]. The adaptor based architecture of the GAT allows different resource allocation managers, such as GRAM and GRMS, to be plugged into Triana. Visual handling of Grid data resources using the GAT is currently being developed within Triana, as are novel techniques for monitoring remote applications (such as the approach outlined in this paper). The aim is to allow users to interact with services running in a Grid environment as seamlessly as if they were running within Triana. Triana is an open source project hosted at Cardiff University [11].

Triana builds upon numerous other workflow technologies — e.g. Triana can networks can be instantiated from BPEL workflow documents, and Triana can export such documents.

3 Cactus

The Cactus Framework [17, 18] is an open source, modular, highly portable, programming environment for collaborative HPC computing. Cactus has a generic parallel computational toolkit with modules providing e.g. parallel drivers, coordinates, boundary conditions, elliptic solvers, interpolators, reduction operators, and efficient I/O in different data formats. Generic interfaces are used, (e.g. an abstract elliptic solver API) making it possible to develop improved modules which are immediately available to the user community. Cactus is used by numerous application communities internationally, including Numerical Relativity e.g. [19, 20], Climate Modelling [21], Astrophysics [22], Biological Computing [23] and Chemical Engineering [24]. It is a driving framework for a number of computing infrastructure projects, particularly in Grid Computing, e.g. GrADS [25], GridLab [26], GriKSL [27], and the ASC [22, 28].

Cactus is distributed with a structured-mesh unigrid MPI parallel driver (PUGH). Other drivers include Carpet [29, 30] (MPI, adaptive mesh refinement (AMR)), and PAGH (MPI/GrACE [31], AMR). Cactus has many features for Grid computing and dynamic scenarios: checkpoint/restart; parameter steering; portability; HTTP interface; output in numerous formats, both binary, e.g. HDF5, and ASCII based, e.g. for use by GNUPlot; and performance monitoring. GridLab is developing modules to integrate Cactus with its Grid Application Toolkit [13, 32], providing any Cactus simulation easy access to a multitude of capabilities, such as job migration or spawning and data replication.

4 Real-Time Visualisation or Analysis of Cactus Simulations within Triana workflows

Within Cactus, users typically want to view or analysis certain files, which can be used to monitor the progress of the application or derive scientific results. Cactus can output data in many formats, such as HDF 5 files, JPEGs, and ASCII formats suitable for visualising with common tools such as X-Graph or GNUPlot. A user would typically want the flexibility of being able to choose, at run time, the files he/she wishes to view or analyse in an interactive fashion. For example, the user may notice from the JPEG images that a simulation of system consisting of two orbiting sources is showing the sources coalescing; this user may then wish to verify these findings by retrieving the detailed simulation data and passing this to other analysis tools or even by converting the output to an audio format and listening to the acoustic waveform directly. Our protocol, therefore, supports the dynamic notification necessary for such interactions. When a file is created, the web service deployed within a Triana unit is notified, and at each time step, the web service is contacted and can choose to receive any of the files that are available. By default the application only sends differences in text files since the last time the web service received part of the file, thus reducing bandwidth; binary files are transferred in entirety. If something interesting happens, the web service can select and receive a different set of files in the next iteration.

This is aided by the use of the Triana solving environment, which allows components to be dynamically added/removed as the application is running.

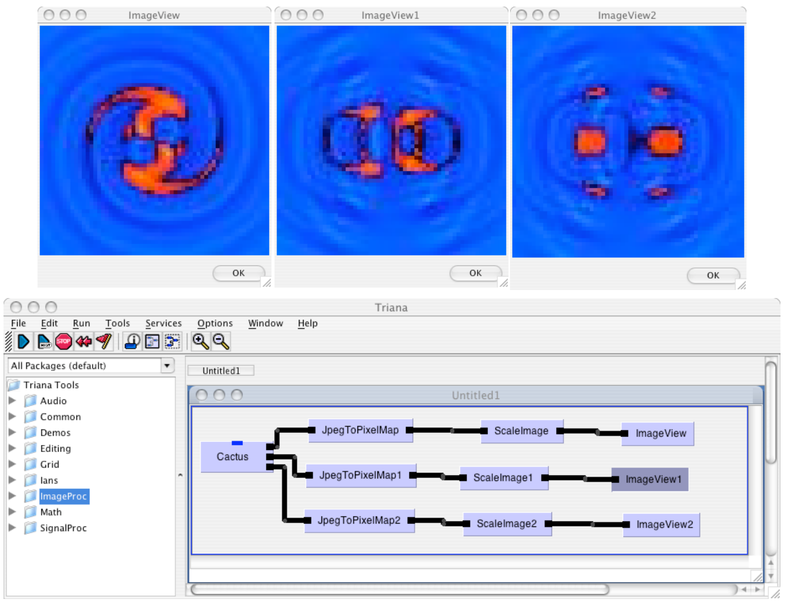

Figure 2: The resulting visualization from a Cactus simulation of the evolutions of a 3D scalar field.

Within Triana, a unit was created to host the Web service representing the underlying protocol. This is shown in Figure 2. The unit upon initialisation uses WSPeer [14] to dynamically create and deploy the Web service within the Axis environment and create the necessary WSDL file representing the methods within the protocol. The actual protocol is quite simple. It involves a notification and selection procedure but it is carefully designed so that it is completely application (i.e. Cactus) driven. This ensures that we do not run into firewall issues (see section 5).

In our initial development, Triana and the Cactus application are deployed and instantiated independently, which is a useful model for occasional monitoring of an application's progress, as it allows a user to make a later decision to use Triana to monitor the output. In the full usage scenario we envisage, however, a Triana unit would also be used to deploy Cactus on a remote resource on demand, thus allowing Triana to manage the full life-cycle of the workflow, as is done in other workflow management systems.

In this current stage of deployment, we are using one Cactus Triana unit per Cactus instance running on the Grid. This approach is not scalable in the visual sense i.e. imagine trying to visualize several thousand Triana units, nor in the networking sense i.e. having thousands of local instances of the same Web service would be impractical for hosting environments. To address these issues, we are currently planning on building a scalable Cactus unit that allows many instances to be mapped internally within one Web service instance. We imagine that this would build around the Triana dynamic scripting or looping implementation (to hide the visual complexity) and then such instances mapped using proxies to a Cactus Triana unit instance for that script or loop. This would allow the connection of possibly hundreds of instances.

If more instances are needed then we can use the Triana distributed mechanisms [33] to segregate the workflow and run it across several Triana GAP services across the Grid, allowing potentially many thousands of instances. However, the algorithmic problem of how these results are analysed would be application specific. Within one scenario involving Cactus, we imagine that Triana would be monitoring the output of its results to see if something interesting had happened (i.e. the apparent horizon of a black hole simulation), then Triana would invoke a separate workflow to farm off many independent Cactus simulations to investigate this phenomenon more closely and then analyse the results upon completion. The user would only wish to view when a certain optimisation level has been reached.

5 Working in Secure Distributed Environments

Traditionally distributed computing protocols have assumed that there are no barriers in the communication between components. In today's internet, however, this is no longer true, as the presence of firewalls provides one-way barriers to communication, and Network Address Translation (NAT) removes even the possibility of creating a connection to some hosts. In the future the deployment of thing such as the Grid Application Toolkit [13] (GAT), or implementations of the Simple API for Grid Applications [34] (SAGA), currently being developed by a research group within the Global Grid Forum (GGF), will again make such infrastructure transparent, as the specialised routing will be hidden in these infrastructure components. Today, however, it is necessary to think carefully when developing a distributed system, and design around firewall issues.

Thus we developed a protocol where all communication is initiated by Cactus, which is the the component which we have the least control of the deployment of, and thus is most likely to be behind a firewall or NAT barrier. Our protocol consists of three parts: notification of the availability of a new file; querying, by Cactus, of the Triana user's desire to see any of these files; and the sending of the files, or the changes in the files.

6 SC2004 Demo and Current Implementation Status

In the SC2004 demonstration, we coded the Triana web-service unit to auto-detect JPEG files output from the Cactus application and represent each one as a separate output node on the unit, as shown in Figure 2. This allows the user to connect cables from these nodes to any of the other Triana units for further analysis or visualisation. There are around 500 Triana units covering a broad range of applications and here we took advantage of the image-processing toolkit to visualise the JPEG files as show. However, a user could reconnect these units during the running the simulation in order to post-process or visualise in a different manner.

Here, the three image windows show the output in the three dimensions. This application is one of the simplest examples of a solving a hyperbolic partial differential equation using finite differences, and so provides a very good learning example of how a PDE can be solved within the Cactus framework. Despite its simplicity, the WaveToy example is prototypical of much more complicated systems of equations.

The complete scenario is illustrated in Figure 3. Here, we can see the Cactus application instigating all of the communication to the client-side visualisation component, thereby avoiding NAT or firewall issues. We also show here, the Cactus steering component, which involves the use of a application-side HTTP server that can be used to receive requests from an off-the-shelf Web browser. Here, obviously there are firewall issues but interestingly this outlines another use of this underlying infrastructure: we imagine a scenario in that users could monitor the application using our protocol and then, based on this information, steer the application accordingly. In this case, our protocol would form part of a larger workflow for generic steering. Within Cactus, this involves the use of the Cactus HTTP server but we are currently discussing extensions that would allow application-driven steering within the the same framework. This would provide a NAT and firewall friendly mechanism to allow applications to be steered but would obviously require application-side interfaces, which could be integrated from elsewhere [35], [13] or [34].

7 Future Work

This work lays the foundations necessary to develop a powerful synergistic suite of Triana units and Cactus thorns which enable the two frameworks to be used in tandem to do high-end, collaborative science, and, in general to speed up the day-to-day research activities of scientists, by allowing Triana to be used as a glue to connect together various simulation code (Cactus in this first case) and analysis tools, such as Triana units or external packages, e.g. scripts, visualisation tools (OpenDX, Amira, IDL, AVS, ...), or other tools, along with any preprocessing of data necessary to start a simulation, for example mesh generation for CFD applications, creation of parameter files, parallel mesh distribution, etc. That is to build the infrastructure in which existing industrial and scientific HPC workflows can be realised within a graphical workflow environment.

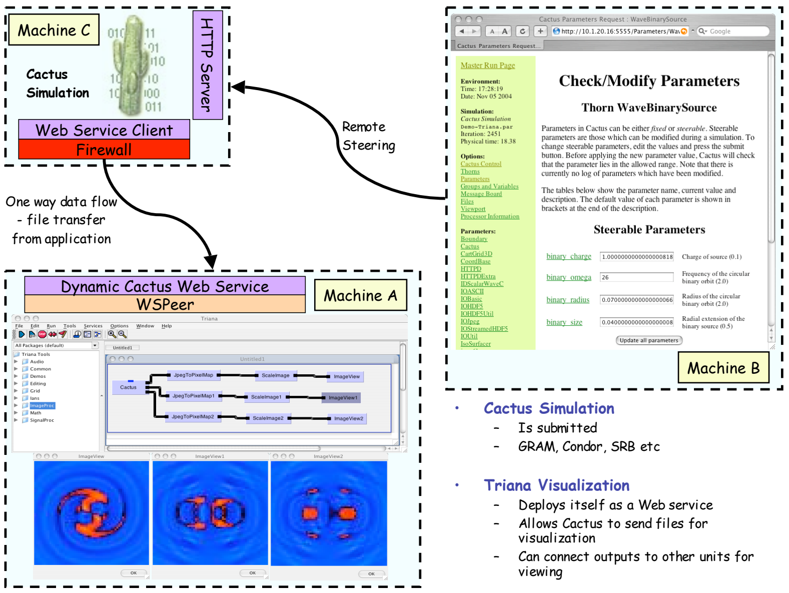

Figure 3: The communication from the distributed legacy application and the client uses a one-way protocols which circumvents firewall issues.

The only key requirement for the use of the protocol described here is that the remote application outputs files. This infrastructure can be extended therefore to be applied to any application that a user might want to interact with in this way. This is what we have achieved in our recent extension to our subsystem that implements this protocol. We have recently written a application-side monitoring tool, which uses the Cactus thorn implementation but rather than triggering directly from the application, it detects when files are output and triggers in this way instead. The user specifies which directories it should monitor and then it discovers files when they are created and notifies the application accordingly.

Using these extensions, the application code does not need to be modified or adhere to any supporting application-side protocol. The application is completely independent of the infrastructure and therefore it should be applicable to many other legacy applications. We are currently in the process of testing other scenarios with our new prototype and when stable, releaser the framework as an open source effort.

As this protocol is independent of the specific tools, Cactus and Triana in this instance, which we have used in this implementation, modules could be developed for other workflow packages, such as SCIRun, ICENI [36], or CCA based frameworks, implementing this protocol and enhancing their ability to interact with independently developed applications.

8 Conclusion

Scientists and engineers implicitly use workflows in their day-to-day work, however these are often not formalised, or require manual execution of the individual components. While there are tools which can make these workflows more explicit and automate them, they are only making slow in-roads into many scientific communities; moreover these tools are rarely Grid aware. In this paper we have presented a simple scheme which allows existing simulation codes to be easily integrated into workflows with little or no modification. By taking two widely deployed tools, Cactus as the simulation tool and Triana as the workflow tool, we have shown how this scheme may be utilised easily, and laid the groundwork for allowing existing Cactus users to make their workflows explicit and make use of the analysis tools which are already available within the Triana framework.

References

[1] CCA Forum. The Common Component Architecture Technical Specification - version 0.5. Technical report, Common Component Architecture Forum, 2001.

[2] S. G. Parker, M. Miller, C. D. Hansen, and C. R. Johnson. An Integrated Problem Solving Environment: The SCIRun Computational Steering System. In *Proceedings of the 31st.˜ Hawaii International Conference on System Sciences (HICSS-31)*, pages 147–156, January 1998.

[3] Condor Home Page `http://www.cs.wisc.edu/condor`.

[4] Scientific Workflows Survey. See web site at: `http://www.extreme.indiana.edu/swf-survey/` .

[5] The open science grid consortium. http://www.opensciencegrid.org/.

[6] The teragrid project. http://www.teragrid.org/.

[7] EGEE: Enabling Grids for E-science in Europe. See website at `http://public.eu-egee.org/`.

[8] Ian Foster and Carl Kesselman. Globus: A Metacomputing Infrastructure Toolkit. *International Journal of Supercomputing Applications*, 11(2):115–128, 1997.

[9] The virtual data toolkit. http://www.vdt.org.

[10] Cactus computational toolkit. `http://www.cactuscode.org`.

[11] The Triana Project. `http://www.trianacode.org`.

[12] K. Czajkowski, I. Foster, N. Karonis, C. Kesselman, S. Martin, W. Smith, and S. Tuecke. A Resource Management Architecture for Metacomputing Systems. In *Proc. IPPS/SPDP '98 Workshop on Job Scheduling Strategies for Parallel Processing*, pages 62–82, 1998.

[13] T. Goodale, K. Davis, and A. Merzky. *GAT API Specification: Object Based.* `http://www.gridlab.org/Software/GAT/Documents/Gridlab-1-GAS-0004.ObjectBasedAPISpecification.pdf`.

[14] Andrew Harrison and Ian Taylor. Wspeer - an interface to web service hosting and invocation. In *HIPS Joint Workshop on High-Performance Grid Computing and High-Level Parallel Programming Models*, 2005.

[15] GEO 600 Project. GEO 600 Homepage, June 2004. See website at `http://www.geo600.uni-hannover.de/`.

[16] Gabrielle Allen, Kelly Davis, Konstantinos N. Dolkas, Nikolaos D. Doulamis, Tom Goodale, Thilo Kielmann, André Merzky, Jarek Nabrzyski, Juliusz Pukacki, Thomas Radke, Michael Russell, Ed Seidel, John Shalf, and Ian Taylor. Enabling Applications on the Grid: A Gridlab Overview. *International Journal of High Performance Computing Applications: Special issue on Grid Computing: Infrastructure and Applications*, 2003.

[17] T. Goodale, G. Allen, G. Lanfermann, J. Massó, T. Radke, E. Seidel, and J. Shalf. The Cactus framework and toolkit: Design and applications. In *Vector and Parallel Processing - VECPAR'2002, 5th International Conference, Lecture Notes in Computer Science*, Berlin, 2003. Springer. `http://www.cactuscode.org/Papers/VecPar_2002.pdf`.

[18] G. Allen, T. Goodale, G. Lanfermann, T. Radke, D. Rideout, and J. Thornburg. *Cactus Users Guide*, 2004. `http://www.cactuscode.org/Guides/Stable/UsersGuide/UsersGuideStable.pdf`.

[19] Apples with apples: Numerical relativity comparisons and tests. `http://www.ApplesWithApples.org`.

[20] EU Astrophysics Network home page. `http://www.eu-network.org/`.

[21] B. Talbot, S. Zhou, and G. Higgins. Review of the Cactus framework: Software engineering support of the third round of scientific grand challenge investigations, task 4 report - earth system modeling framework survey. `http://sdcd.gsfc.nasa.gov/ESS/esmf_tasc/Files/Cactus_b.html`.

[22] Astrophysics Simulation Collaboratory (ASC) home page. `http://www.ascportal.org`.

[23] Illinois BioGrid. `http://www.illinoisbiogrid.org/`.

[24] K. Camarda, Y. He, and K. A. Bishop. A parallel chemical reactor simulation using Cactus. In *Proceedings of Linux Clusters: The HPC Revolution, NCSA*, 2001. `http://www.cactuscode.org/Papers/Camarda01.doc`.

[25] G. Allen, D. Angulo, I. Foster, G. Lanfermann, C. Liu, T. Radke, E. Seidel, and J. Shalf. The Cactus Worm: Experiments with dynamic resource discovery and allocation in a grid environment. *Int. J. of High Performance Computing Applications*, 15(4), 2001. `http://www.cactuscode.org/Papers/IJSA_2001.pdf`.

[26] GridLab: A Grid application toolkit and testbed project home page: `http://www.gridlab.org`.

[27] DFN-Verein project *"development of grid based simulation and visualization techniques"* (GRIKSL) home page. `http://www.griksl.org`.

[28] R. Bondarescu, G. Allen, G. Daues, I. Kelley, M. Russell, E. Seidel, J. Shalf, and M. Tobias. The Astrophysics Simulation Collaboratory portal: a framework for effective distributed research. *Future Generation Computer Systems*, 2003. Accepted. `http://zeus.ncsa.uiuc.edu/∼ruxandra/asc.pdf`.

[29] E. Schnetter, S. H. Hawley, and I. Hawke. Evolutions in 3D numerical relativity using fixed mesh refinement. *Class. Quantum Grav.*, 21(6):1465–1488, 21 March 2004. `http://arxiv.org/pdf/gr-qc/0310042`.

[30] Fixed mesh refinement with Carpet. `http://www.tat.physik.uni-tuebingen.de/∼schnette/carpet/`.

[31] Grid Adaptive Computational Engine (GrACE) `http://www.caip.rutgers.edu/~parashar/TASSL/Projects/GrACE/`.

[32] Grid Application Toolkit web-page. `http://www.gridlab.org/WorkPackages/wp-1`.

[33] Ian Taylor, Matthew Shields, Ian Wang, and Omer Rana. Triana Applications within Grid Computing and Peer to Peer Environments. *Journal of Grid Computing*, 1(2):199–217, 2003.

[34] SAGA Research Group. See web site at: `https://forge.gridforum.org/projects/saga-rg/`.

[35] Reality grid project. http://www.realitygrid.org/.

[36] S. McGough, L. Young, A. Afzal, S. Newhouse, and J. Darlington. Workflow Enactment in ICENI. In *UK e-Science All Hands Meeting*, 2004.